https://dx.doi.org/10.22161/ijaers.4.4.3 ISSN: 2349-6495(P) | 2456-1908(O)

Research on Multiple Complex Data Processing Methods Based on OpenStack Cloud Platform

Huansong Yang, Mengyuan Wang, Jiaping Wu

Abstract— *OpenStack is an open source cloud computing management platform project that supports almost all types of cloud environment. It can achieve data processing services among the interactive information storages, and it can also be stored in the virtual machine of cloud computing platform in various services. When performing complex data combination processing, each service cooperates with other services according to the interaction information, and finally completes the processing of complex data.*

Keywords— OpenStack; Complex data processing, Cloud computing, Service.

I. INTRODUCTION

OpenStack is an open source cloud computing management platform project that supports almost all types of cloud environments and helps service providers and enterprises implement infrastructure as a service (IaaS) similar to Amazon EC2 and S3. The OpenStack cloud platform contains several key projects, including Compute, Identity Service, Networking, Image Service, Block Storage, Object Storage, Telemetry, Orchestration and Database, which can be installed independently and deployed on demand. People can install any of these projects independently, provide independent services through configuration, or communicate with other projects to form a feature-rich and powerful cloud service system.

II. DESIGN OF MULTIPLE COMPLEX DATA PROCESSING

After the study of OpenStack, OpenStack can be achieved based on a number of complex data processing methods, as follows, as shown in Figure 1.

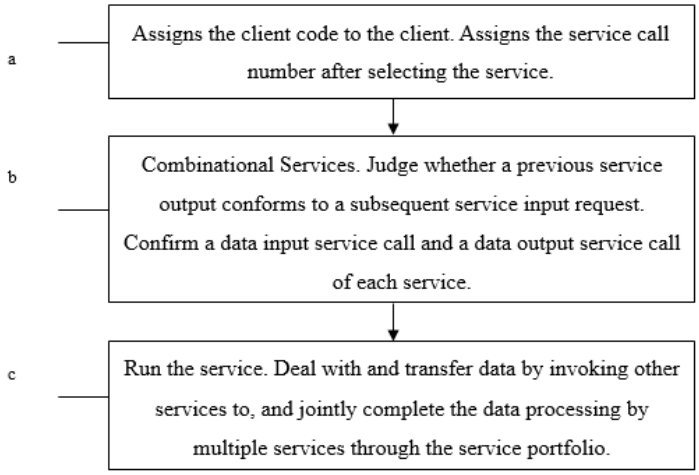

Fig.1: OpenStack based on multiple complex data processing flow chart

(a) Assign a client number to the client. Select the desired service from the service library, and then assign a service call number to this selection for the service.

(b) Combination of selected services. If the former of two services connected to is the data output service call, the latter is for the data input service call, which contains the following process:

(b1) Using the data matching judgment unit to estimate whether the two service output types and the input requirements match, if succeed, then (b3), otherwise the

https://dx.doi.org/10.22161/ijaers.4.4.3 *ISSN: 2349-6495(P) | 2456-1908(O)*

combination is not successful;

(b2) For data output service calls, insert a row of data in the service composition structure database，including: customer number (number of customers assigned for this service composition function), service call number (number of data output service calls, uniquely identifying the service), the next service address (the service address of the data input service call, exposing the service to other customers or services using the address in the form of a network application interface (WEB API)), the next service call number (number of data entry service calls).

(c) Run the service. Complete the data processing through the way of service combination. For any one of the running of service, it includes the following specific process:

(c1) Receive the data of the previous service, including the next service call number, the customer number parameter, and the data to be processed;

(c2) Processing the received data, completing the data processing function of the service, and outputting the processed data;

(c3) Find the customer number obtained by the Customer Number field in (c1) through the Service Structure database, the service call number field is the line of the next service call number obtained in (c1), To obtain the service after the next service call number and service address, through the service address call the next service, the transfer of the service data, but also includes the customer number, the service after the next service call number.

III. SOLUTION OF MULTIPLE COMPLEX DATA PROCESSING

(a) Assign unique client number C_i to the client, and assign a service call number S_{j,C_i} for each service selected by the client;

Where C_i is the number assigned by the i-th client, S_{j,C_i} is the service call number assigned to the j-th service assigned by the client with the client number C_i, the corresponding service for the sensor data after processing to get the final road traffic data in the process of a sub-processing process;

(b) During the process of selecting a combination, invoke S_{j,C_i} and S_{j+1,C_i} for any two associated connected services respectively correspond to two data processing sub-processes in the traffic data processing process. Among them, comparing with S_{j+1,C_i}, S_{j,C_i} processes output service call for data, and requirement of the output type is $T\_S_{j,C_i}(\text{oarg}_1, \text{oarg}_2, \text{oarg}_3, ..., \text{m})$. Comparing with S_{j,C_i}, S_{j+1,C_i} processes input service call for data, and requirement of the input type is $\text{R\_S}_{j+1,C_i}(\text{inreq}_1, \text{inreq}_2, \text{inreq}_3, \ldots, \text{n})$. The data matching judgment unit is used to tell whether $T\_S_{j,C_i}(\text{oarg}_1, \text{oarg}_2, \text{oarg}_3, ..., \text{m})$ and $\text{R\_S}_{j+1,C_i}(\text{inreq}_1, \text{inreq}_2, \text{inreq}_3, \ldots, \text{n})$ match or not. If the match succeed, the service invokes the row of data in the service composition database of the server where the S_{j,C_i} is located, including the customer number C_i, the data output service call number S_{j,C_i}, the data input service call address $URL(S_{j+1,C_i})$, and the data input service call number S_{j+1,C_i}.

Where the $T\_S_{j,C_i}(\text{oarg}_1, \text{oarg}_2, \text{oarg}_3, ..., \text{m})$ is the type of the K parameter of the service call TT for the service, m is the number of parameters, and $\text{inreq}_t(\text{t} = 1, 2, 3, ..., \text{t} \le \text{n})$ is the type of the t th input parameter of the S_{j+1,C_i} call for the service in the $\text{R\_S}_{j+1,C_i}(\text{inreq}_1, \text{inreq}_2, \text{inreq}_3, \ldots, \text{n})$;

https://dx.doi.org/10.22161/ijaers.4.4.3 *ISSN: 2349-6495(P) | 2456-1908(O)*

$URL(S_{j+1,C_i})$ call the remote service program used by the address for the use of network application program technology.

(c) Run the service. Through the method of service portfolio, jointly complete the massive data processing with the combination of services. The process is shown as follow.

(c1) For any service call number S_{j,C_i} for the service operation, people need to receive the previous service call S_{j-1,C_i} data. The data includes the service call number S_{j,C_i}, the customer number C_i and the data to be processed $Data$. And then process data to complete the service data processing function, obtaining the processed data $RS(\text{Data})$.

(c2) Find the customer number C_i via the service structure database service call number field S_{j,C_i}, so that to obtain the service after the next service call number S_{j+1,C_i} and service address $URL(S_{j+1,C_i})$. Then call the next service through the service address $URL(S_{j+1,C_i})$, and pass S_{j,C_i} processing Data $RS(\text{Data})$, customer number C_i and next service call number S_{j+1,C_i}.

Further, in the step (b), the process of the data matching is: firstly, judging whether m or n satisfies $m = n$. If it is not satisfied, the matching does not succeed. Otherwise, secondly judging whether $o\arg_k, inreq_t (k = t, 且 k, t = 1, 2, 3...)$ satisfies $o\arg_k = inreq_t$ or not. If it satisfies, the match succeeds. Otherwise the match does not succeed.

IV. METHODS OF MULTIPLE COMPLEX DATA PROCESSING

Multiple complex data processing process implementation methods based on OpenStack are as follows:

1. (1) Pick a number of required services from the service list and assign call code to each selected service. For example, sx（x=1，2，3…）is the service number, call_id_y (y = 1, 2, 3 ...) is the service call number. If two or more service numbers are the same service, the input data source and the output destination service processed in the combination are different. There will be the same service number, but different service call number. It is shown in Figure 2.

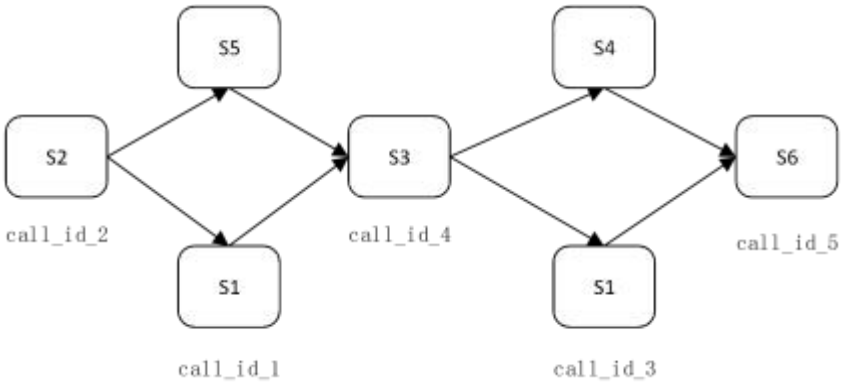

Fig.2: Service call

As shown in Figure 2, s1 is the case. Call the s1 service with the number call_id_1 whose data input source is the s2 service calling the call_id_2, the data output is the s3 service calling the call_id_4 number. And call the s1 service with the number call_id_3, the input data source is called s3 service with the number call_id_4, and the data output is

https://dx.doi.org/10.22161/ijaers.4.4.3 ISSN: 2349-6495(P) | 2456-1908(O)

called s6 service with the number call_id_5. These two identical s1 service calls have the same functionality, but belong to different service calls, so the service call number is different.

(2) Assign the customer number user_id_z (z = 1, 2, 3...), in the service where the server, a service can be used by multiple customers can also be used by the same customer several times, resulting in different service calls. When customer selects a service, the service call number is independent. When different customers choose a service the service numbers may be the same. Therefore, in the server, when different customers call the same service, people can distinguish different service calls through the combination of the customer number and service call number.

For example, s2 is used three times by two clients (user1_id, user2_id): the user with the number user1_id is used twice to form two service calls calling call_id_1 and call_id_2. The number of user2_id customers used once, forming a call number call_id_1 service call. At this point, s2 of two service call numbers are call_id_1. But they belong to different service calls for different clients, so these two different service calls are distinguished by user1_id + call_id_1 and user2_id + call_id_1.

2. The service side of each service needs to maintain a database of a service composition structure. The database has four fields: user_id, call_id, next_call_id and next_service_url:

User_id: Call the customer number of the service, along with the call_id field, is used to distinguish the service calls to which the service belongs;

Call_id: A service call number for this service, used in conjunction with the user_id field to distinguish service calls to which the service belongs;

Next_call_id: After the end of this service, you need to call the next service call number, in the next service, is used with the user_id to distinguish the next service call；

Post service_url: The address of the next service call, after the end of this service，the network application program interface is used to call the next service；

In the service portfolio structure database, complete the above data modification and insert operation during the process of customer combination. In the s1→s2→s3service composition process, each service composition client has a number. Assume that the current client assigns the number: user1_id, S1 service call number is call_id_1, service number is s1_id；S2 service call number is call_id_2, service number is s2_id；S3 service call number is call_id_3, service number is s3_id.

(1) When s1 and s2 are combined, the following events are triggered:

Through the s1 service number s1_id query service library to obtain s1 input data requirements s1_input_type, Output data type s1_output_type, service address s1_url;

Through the s2 service number s2_id query service library to obtain s2 input data requirements s2_input_type, Output data type s2_output_type, service address s2_url;

The data matching judgment unit judges whether the data output type of the S1 and the data input type of the s2 match, and if matched, the following operation is performed:

For the s1 service structure database, insert a row of data, among them, the user_id field is user1_id, the call_id field is call_id_1, the next_call_id field is call_id_2, and the next_service_url field is s2_url.

https://dx.doi.org/10.22161/ijaers.4.4.3 ISSN: 2349-6495(P) | 2456-1908(O)

If not, the combination of s1 and s2 fails.

(2) When using s2 and s3 combination, the trigger event is as follows:

Through the s2 service number s2_id query service library to obtain s2 input data requirements s2_input_type, Output data type s2_output_type, service address s2_url.

Through the s3 service number s3_id query service library to obtain s3 input data requirements s3_input_type, Output data type s3_output_type, service address s3_url.

The data matching judgment unit judges whether the data output type of the S1 and the data input type of the s2 match, and if matched, the following operation is performed:

For the s2 service structure database, insert a row of data, among them, the user_id field is user1_id, the call_id field is call_id_2, the next_call_id field is call_id_3, and the next_service_url field is s3_url.

If not, the combination of s2 and s3 fails.

Three basic combinations of services and the operation of the process:

Single input single output form, as shown in Figure 3:

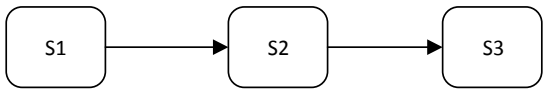

Fig.3: Single-input single-output form of service portfolio

Service structure database where service s2 the server maintenance has a row of records, indicating that service output invoked by service invocation is only s3. Assume that the current client assigns an id of: user1_id, s1 service call id is call_id_1, s2 service call id is call_id_2, and s3 service call id is call_id_3.

S2 server in the service structure of the database has a record as shown in Table 1:

Table.1: Structure data record table

user_id	call_id	next_call_id	next_service_url
user1_id	call_id_2	call_id_3	s3_url

The entire calls are as follows:

S1 at the end of data processing, send s1's next service service call number call_id_2, customer number user1_id and source data data1 to the service side of service s2;

S1 accept the original data s2 sent over, and s1 service of the next service call number call_id_2, client number user1_id, s2 processing data1 to get the processed data data2;

Service s2 to find its server-side service structure database, find the call_id for call_id_2, user_id for all user1_id, so as to obtain a call URL for s3_url, and a service call number call_id_3, and through service to address s3_url, call s3 in the form of WEB API, and send data data2 and user number user1_id, the next service call number call_id_3.

Single input multiple output form is shown in Figure 4.

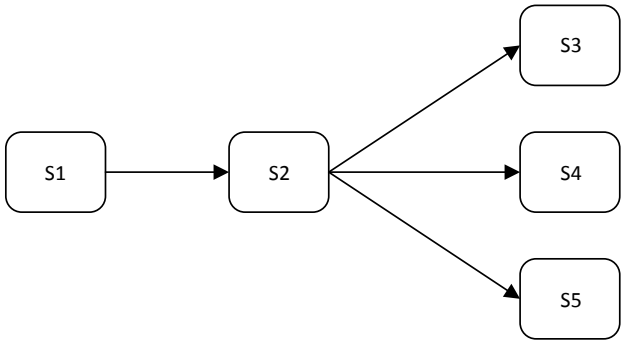

Fig.4: Single-input multi-output form of service portfolio

Service s2 where the service side of the server to maintain

https://dx.doi.org/10.22161/ijaers.4.4.3 ISSN: 2349-6495(P) | 2456-1908(O)

the service structure of the database has multiple lines of records, indicating that the service call has multiple calls to the service output. Assume that the current client assigns the number: user1_id. The service call number for s1 is call_id_1. The service call number for s2 is call_id_2. The service call number for s3 is call_id_3. The service call number for s4 is call_id_4. The service call number for s5 is call_id_5.

S2 server in the service structure of the database is shown in Table 2.

Table.2: Service structure data records

user_id	call_id	next_call_id	next_service_url
user1_id	call_id_2	call_id_3	s3_url
user1_id	call_id_2	call_id_4	s4_url
user1_id	call_id_2	call_id_5	s5_url

The entire calls are as follows:

S1 at the end of data processing, send s1's next service call number call_id_2, customer number user1_id and source data data1 to the service side of service s2;

S1 accepts raw data sent by s2, and s1 service of the next service call number call_id_2, client number user1_id, s2 processing data1 to get the processed data data2;

Service s2 to find its server-side service structure database, find the call_id for call_id_2, user_id for all user1_id, so as to obtain a call URL for s3_url, s4_url and s5_url. As well as the corresponding service call number call_id_3, call_id_4, call_id_5, through the service address s3_url, s4_url, s5_url, in the form of network application program interface (web api), respectively call the service s3, s4, s5 and send data data2 and customer number user1_id, and The corresponding next service call number.

Multi-input single output form is shown in Figure 5:

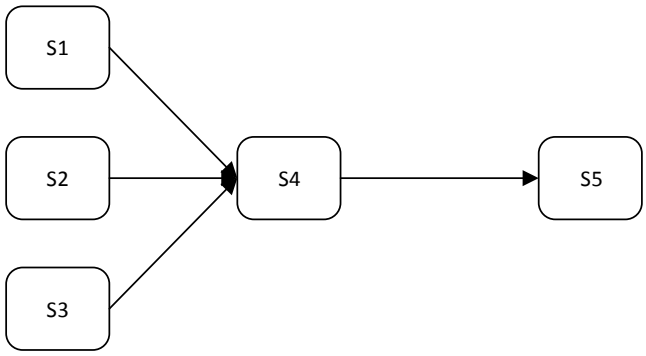

Fig.5: Multi-input single-output form of service portfolio

The service structure database maintained by the S2 server has multiple lines of records, but the output service call is only one. Assume that the current client assigns the number: user1_id, call the S1 service call number call_id_1, S2 service call number call_id_2, S3 service call number call_id_3, S4 service call number call_id_4, S5 service call number is call_id_5.

S2 server in the service structure of the database shown in Table 3:

Table.3: Service structure data records

user_id	call_id	next_call_id	next_service_url
user1_id	call_id_4	call_id_5	s5_url

https://dx.doi.org/10.22161/ijaers.4.4.3 ISSN: 2349-6495(P) | 2456-1908(O)

The entire call processes are as follows:

S3 at the end of the data processing, the next service call number is call_id_4. The client number user1_id and the source data datax are passed to the server of the service s4. (x = 1, 2, 3);

S4 accepts raw data sent by sx,, and sx service of the next service call number call_id_4, client number user4_id, s4 processing datax to get the processed data data4; (x = 1, 2, 3);

Service s4 to find its server-side service structure database, find the call_id for call_id_4, user_id for all user4_id, so as to obtain a call url for s5_url, and a service call number call_id_5, and through service to address s5_url, call s5 in the form of WEB API, and send data data4 and user number user1_id, the next service call number call_id_5.

Because the input data of the service is passed through the call of the previous service, all the input sources of the service are single input. The operation of the s4 service is not running under s1, s2, s3, but by s1, s2, s3 call each, so this multi-input single-input mode is equivalent to s1 → s4 → s5, s2 → s4 → s5 and s3 → s4 → s5 combination.

Other forms of combination are formed by the above basic combination. And the final data processing is jointly completed by multiple service combinations.

V. CONCLUSION

The above-mentioned OpenStack complex data processing solution can realize the interactive information storage between data processing services and can be stored in the virtual machine of cloud computing platform where each service is located. When the complex data combination processing is performed, each service in accordance with the interaction of information and other services to coordinate, and ultimately together to complete the processing of complex data. As a client, the required resources can be quickly obtained and the services can be quickly deployed and provided to other users in the form of services. Other users can use the service composition technology to complete the complex data processing process.

REFERENCE

[1] 北京航空航天大学. 一种面向云制造的服务组合路径构造方法:中国, 10011043,[P].2015-01-13.

[2] Y.Xia, M.Zhu. Power-aware small world topology in ad hoc networks. Journal of Computers,7(1),Jan. 2012.

[3] M. Sharma, P. Bedi, K. Chaturvedi, V. Singh, Predicting the priority of a reported bug using machine learning techniques and cross project validation, in: Proceedings of the 12th International Conference on Intelligent Systems Design and Applications, ISDA '12, 2012, pp. 539-545.

[4] Y. Tian, D. Lo, C. Sun, Drone: Predicting priority of reported bugs by multi-factor analysis, in: Proceedings of the 29th IEEE International Conference on oftware Maintenance, ICSM '13, 2013, pp. 200-209.

[5] Y.Xia, Y.Liu, Z.Ye, W.Wu, M.Zhu. Quadtree-based domain decomposition for parallel map matching on GPS data. Proceeding of 15th IEEE Intelligent Transportation Systems Conference (ITSC 2012), Anchorage, AK, Sep. 2012.

[6] 廖军, 谭浩, 刘锦德. 基于 Pi-演算的 Web 服务组合的描述和验证[J]. 计算机学报, 2005, 28(4): 635-643.

https://dx.doi.org/10.22161/ijaers.4.4.3 *ISSN: 2349-6495(P) | 2456-1908(O)*

[7] OpenStack is the open alternative to proprietary cloud platforms and lock-in，http://www.rackspace.com/cloud/openstack/

[8] XCP Home Page [EB/ OL]. http://www.xen.org/products/cloudxen.html, 2011.

[9] Eucalyptus project [EB/ OL]. Eucalyptus Administrator's Guide, 2011.

[10] OpenStack Nova Architecture, http://ken.pepple.info/openstack/2011/04/22/openstack-nova-architecture/

[11] Dynamic service composition: state of the art and research directions[R], Department of Computer Science and Electrical Engineering, University of Maryland, USA, http://w w w.cs.umbc.edu/～dchakr1/papers/techreportcomposition.ps, 2001.

[12] Xiaolong Wen; Genqiang Gu; Qingchun Li; Yun Gao; Xuejie Zhang. Comparison of open-source cloud management platforms: OpenStack and OpenNebula. Fuzzy Systems and Knowledge Discovery (FSKD), 2012, 9(2457 – 2461).

[13] F. Wuhib, R. Stadler, and M. Spreitzer, “A gossip protocol for dynamic resource management in large cloud environments,” Network and Service Management, IEEE Transactions on, 2014.